Efficient Web Browsing with a Single-Switch

Laurianne Sitbon
Queensland University of Technology
Brisbane, Australia
l.sitbon@qut.edu.au

Oscar Wong
Queensland University of Technology
Brisbane, Australia
oscarwong91@gmail.com

Margot Brereton
Queensland University of Technology
Brisbane, Australia
m.brereton@qut.edu.au

ABSTRACT

This paper presents a new approach to web browsing in situations where the user can only provide the device with a single input command device (switch). Switches have been developed for example for people with locked-in syndrome and are used in combination with scanning to navigate virtual keyboards and desktop interfaces. Our proposed approach leverages the hierarchical structure of webpages to operate a multi-level scan of actionable elements of webpages (links or form elements). As there are a few methods already existing to facilitate browsing under these conditions, we present a theoretical usability evaluation of our approach in comparison to the existing ones, which takes into account the average time taken to reach any part of a web page (such as a link or a form) but also the number of clicks necessary to reach the goal. We argue that these factors contribute together to usability. In addition, we propose that our approach presents additional usability benefits.

Keywords

disability, web browsing, single switch, scanning

Categories and Subject Descriptors

H.5.2. [**Information Interfaces and Presentation (e.g. HCI)**]: User Interfaces

1. INTRODUCTION

Internet is central in many people's lives, as much as a tool for accessing services, as a source of knowledge, entertainment and communication. Modern websites are generally navigated through by a pointing device (e.g. mouse) and a keyboard. Although there are some efforts put on creating new browsing experience (e.g. Opera face gestures), a mouse and a keyboard are still the devices that websites are designed for. While keyboards are generally involved to inform text fields in forms, pointing devices are used to activate contents such as hyperlinks, form elements, but also display of dynamic contents.

OZCHI '14, Dec 2-5, 2014, Sydney, Australia

ACM 978-1-4503-0653-9/14/12 ...$15.00
http://dx.doi.org/10.1145/2686612.2686693.

While people are trying to improve the browsing speed by introducing different gestures, one should not ignore people with motor impairments. Some people with physical disability cannot manipulate classical pointing devices or keyboards, which is why single switch inputs have been developed and are often used to control virtual keyboards and desktop based software with a minimal physical requirements. In these instances desktop software (including virtual keyboards) is designed or augmented to integrate a scanning mechanism, which sequentially highlights different actionable elements of the software for selection by the user. Other technologies specially designed as a replacement for a mouse as pointing device include eye tracking devices, however despite their high efficiency ([3] suggested that it can be 60% faster than using mouse control), they remain very expensive and are therefore not accessible to all.

In order to reduce the barrier to access, we propose a novel light-weight web browser plugin which enables users to web-surf using a single switch, by dynamically generating a scanning layer on webpages using a browser plugin technology. In this paper, we first present our methodology that leverages the hierarchical structure of web pages to facilitate access to specific parts of the page, in particular links. We then present a theoretical evaluation of usability to compare our approach to existing ones, using simulation. Namely, we measure the average wait time and the average number of clicks necessary to attain any link in any page.

2. RELATED WORK

Although there are some single-switch browsing techniques being developed, only a few of them are designed for web browsing. They generally follow one of two navigation algorithms as follows.

2.1 Tabbing

Tabbing is the most common approach for single-switch navigation. In webpages, most of the elements can be navigated through pressing the tab key on the keyboard. Some of the applications, for example AVANTI browser [5], MultiWeb Browser [2] and the Hawking toolbar extension for Mozilla Firefox, simulate the "tabbing" action on keyboard and scans through elements at a fixed time interval. There are other variations which extract all the links and display them on a separated list [1]. In both cases, the user accesses the active zones of the page (links and form elements) in sequential order. For example, with a time interval of 1 sec-

ond, it would take 31 seconds to reach the link to the second page of results for a Google search for OzChi2014. Tabbing implicitly allows scrolling control of a page, as the display will start adjusting to new contents to ensure a currently selected tab is always visible. It is an advantage (provided the user can control the activation of scanning) as actionable zones are the ones of current interest. However it also means that the user is limited to the pace of the scan to read contents, which in turns depends on the number of actionable zones presented on the screen.

2.2 Keyboard Navigation

A more efficient alternative keyboard navigation. Virtual keyboards are widely available and generally offer single switch on-screen scanning capabilities. A number of optimisations of the layout, including hierarchical layouts, accelerate access to a given key (rather than offering sequential access).

The web browser Conkeror (http://conkeror.org), developed based on Mozilla Firefox, assigns a navigation key (ID) to each elements on a web page. Another example is the Keysurf plugin on Mozilla Firefox, which is a search based navigation system with a ranking system for the result elements in the page [4]. Keysurf also offers access via a unique naming of each element, which can be a combination of letters and numbers where the letters are generally the first letters of the element (for links) and numbers follow in case several element have the same starting letter. Non lexical elements (text boxes or images) are represented with numbers only. For example, for a document with 3 link named *apple*, *april* and *amazon*, all the links started with *a* can be selected by typing *a* , *a0* and *a1*, instead of *ap1*, *ap2* and *am*. Keysurf is the most well-developed plugin and has demonstrated state of the art performance in terms of the accuracy and key-stroke efforts.

Keyboard navigation requires the user to fully control scrolling of the page displayed, which requests additional input from the user and can considerably increase the time spent by a user in controlling the view in addition to accessing actionable zones.

3. THE 1CLICK BROWSING APPROACH

Our single input web browsing plugin is a Google Chrome plugin which uses a "divide and conquer" approach, leveraging hierarchical information on the display of contents presented in web pages.

When a page is loaded, each high level zone in the page is highlighted in a scanning sequence with a fixed time interval.A highlighted zone is the current active zone which can encompass content as well as actionable zones. When the user clicks on an active zone, if this zone contains several lower level content zones, then these are sequentially highlighted in turn. If the active zone contains only one element that is an actionable zone, then the click equates to that of a regular pointing device click. An example of sequence of scanning and clicks is presented in figure 1.

When a zone is highlighted, the position of the page is automatically updated. That is, there is no need for the user to scroll down. [1]

[1]Additional single input controls have been integrated in the plugin to allow the scrolling to pause and to reverse actions. However these are not the focus of the present paper.

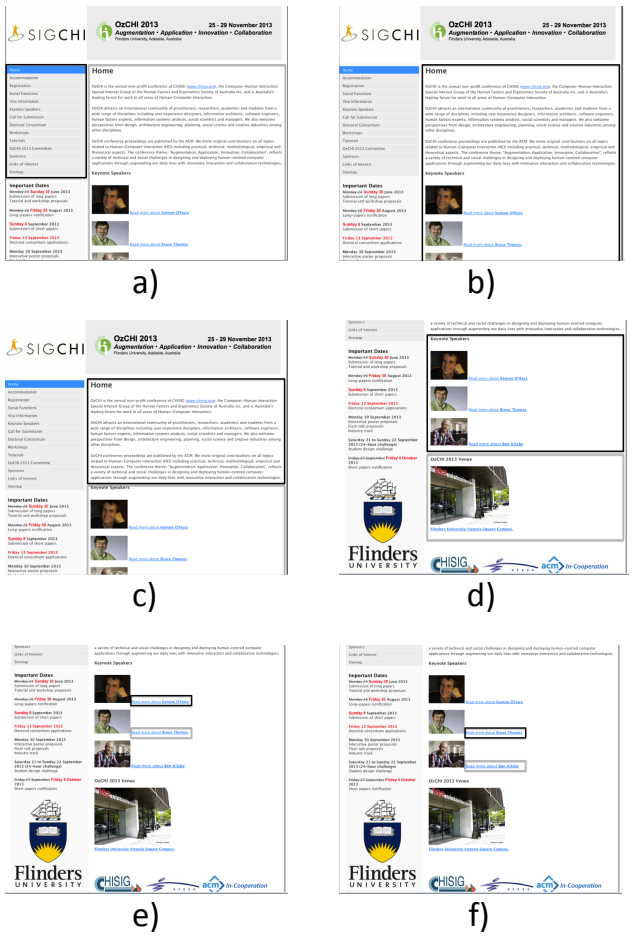

Figure 1: Example of successive views to attain a specific link on the OzChi 2013 webpages. Clicks occur after steps b), d) and f). The active zone is represented in black and the future active zone in grey.

A green border is added to indicate the next element in the scanning sequence (represented in grey on figure 1) in order to support anticipation, which is a typical issue of scanning approaches ([4]).

From a technical perspective, the algorithm first builds a hierarchy of all the elements on the page and stores it in a data structure. The hierarchy uses HTML tags in the same fashion that they are reconciled with CSS. The representation is then compressed to remove the empty layers. Empty layers are common for websites that use CSS for display, and if not compressed and can result in additional delays to reach an element. This is why the entire mapping has to be done completely when the page is loaded rather than dynamically as the user selects subsections. Figure 2 shows an example of a final representation.

4. SIMULATION AND EVALUATION

We propose that the two measurable factors affecting usability of such a system are waiting time (WT) and number of user actions (NUA). Waiting time is proportional to the

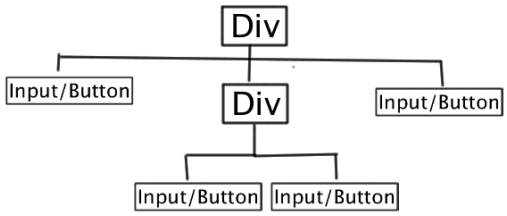

Figure 2: Example of compressed structure obtained with the mapping algorithm

number of zones (single elements or groups) the scanner traverses to arrive at the desired element in a page, whether it is scanning directly on the page, through a list of actionable zones, or on a virtual keyboard. The number of user actions is the number of times the user actions the switch to reach a given actionable zone. While wait time may lead to frustration and lack of patience, excessive number of necessary clicks may lead to fatigue.The best balance between these two factors may also depend on the scanning speed that a user may be able to work with. If the scan is fast, then a high number of clicks would be more tiring and and differences in wait time may not be very significant. On the contrary, if the scan is slow, then more clicks would be preferred to very long wait times.

We first establish the estimated average performance on these two factors for our 1Click plugin, Keysurf, and the linear scanning navigation. This theoretical measure uses one or all of structure of the documents (for 1Click), the number of elements in the documents (for scanning) or the lexical organisation of the elements (for KeySurf).

We next simulate the behaviour of these 3 methods to measure their average NUA and WT on actual websites. Our evaluation is based on the 121 most popular websites as listed on Alexa in January 2013. It is a combination of the top 100 viewed websites worldwide and in Australia with duplicates under various domain names removed. Search engines are evaluated at the first search results page. While several of these websites are in non alphabetical languages not suitable keyboard-based approaches, we include them in the evaluation as they do represent a majority of users worldwide but also in multicultural Australia.

4.1 Tabbing

Tabbing navigation is a linear scanning operation, which only involves one user action when the target is reached. The theoretical average wait time to access any element can be expressed by the following function:

$$AWT = \frac{\sum elementwaittime}{Total\ number\ of\ elements} \quad (1)$$

4.2 KeySurf

We estimate and simulate KeySurf efficiency in a single-switch context with the code-based access and a dynamic virtual keyboard, which groups characters into tiers according to their appearing frequency in the document, as illustrated on Figure 3 where 1st tier would be the 2 most frequent letters in the document (eg. e and w). Hierarchical scanning scrolls over lines first and then individual characters (including letters, numbers and commands). In this design, every key can be attained in a maximum of 6 wait time

Enter	1st tier	2nd Tier	3rd Tier	4th Tier	5th Tier	6th Tier	scroll up
1st tier	2nd Tier	3rd Tier	4th Tier	5th Tier	6th Tier		
2nd Tier	3rd Tier	4th Tier	5th Tier	6th Tier			
3rd Tier	4th Tier	5th Tier	6th Tier				
4th Tier	5th Tier	6th Tier					
5th Tier	6th Tier						
6th Tier							
scroll down							

Figure 3: Virtual keyboard to simulate KeySurf

periods while reducing this time for the most needed keys. KeySurf requires a code composed of one letter or number, and/or a number with a number of digits depending on the number of actionable elements likely to start with the same letter. As an estimate, we can consider that any component of the code can be reached in 3.5 wait time period on the virtual keyboard.

In addition, this approach requires a scroll up and scroll down button to ensure that the user can visualise all the eligible elements if a single browser frame cannot display all the elements. By a rough estimation, the maximum number of elements in a single browser frame is around 200. It means that if the document has around 200 -400 elements, the user has to scroll once to select all the elements in the document. This value is just for reference however it does affect the usability of the extension.

The theoretical average wait time for KeySurf cannot be exactly determined as it depends on the distribution of the letters of the actionable zones and their nature, however we propose to estimate it as follows:

$$AWT = 7*EWT*|\frac{TNE}{200}|+3.5*(1+LOG_{10}(1+\frac{TNE}{26})) \quad (2)$$

where EWT is the individual element wait time and TNE is the total number of elements. The first part of the equation is for the estimated number of scrolls, and the second part is for the estimated number of letters/numbers of the code.

4.3 1Click browser plugin

With our plugin, WT and NUA depend on the shape of the tree that maps the structure of the document. The depth of an element in the tree is the NUA needed for that element (user input is required to go from a higher to a lower level in the structure). The wait time depends on the width of the lower subtree containing the element, and on that of all his parents.

An estimate can therefore be drawn from the average width and depth of the compressed document structure:

$$AWT = \frac{Avg.\ Width\ of\ Layers}{2 * Avg.\ depth\ of\ structure} \quad (3)$$

$$ANUA = Avg.\ depth\ of\ structure \quad (4)$$

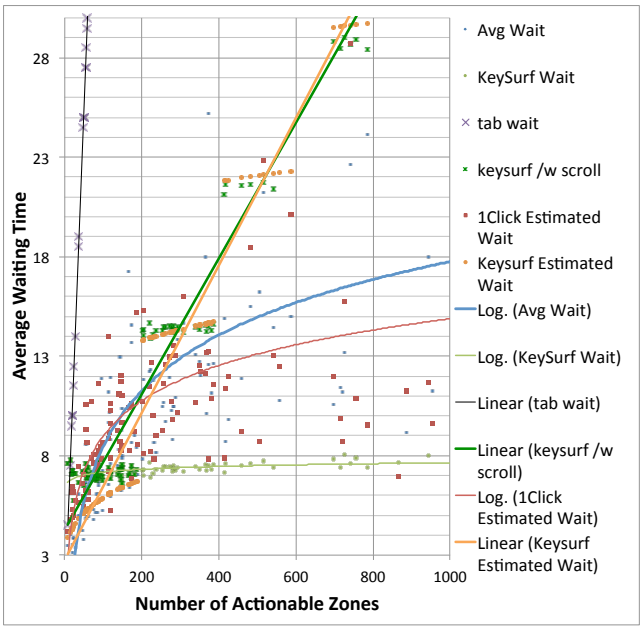

Figure 4: Average wait to reach any link for each approach (estimated, simulated and interpolated).

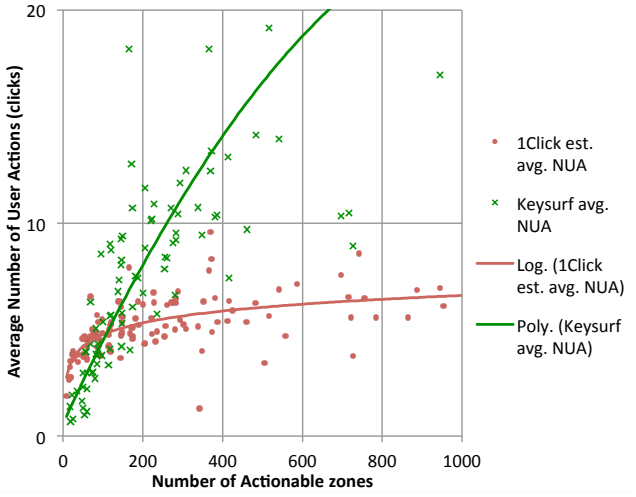

Figure 5: Average number of user action (NUA) for 1Click and KeySurf (tab is always 1).

5. RESULTS

The average WT have been estimated and simulated for each approach and reported based on the number of elements for each of the 121 most popular web pages in Figure 4. In combination with the proposed dynamic virtual keyboard and not accounting for scrolling, KeySurf has a consistent performance between 6 and 8 units of time, including for small documents, making 1Click a better choice for documents under 100 elements. If we estimate a need to scroll every 200 elements, then the AWT increases rapidly. If scrolling controls were placed at the beginning of the keyboard, then they would incur an additional wait time for small pages as well. In this context, the 1Click Browser plugin show better performances in terms of average wait time also for larger documents.

The estimated (theoretical) AWT and simulated AWT for the 121 most popular web pages, as well as their interpolations presented on Figure 4 shows a reasonable alignment although the estimates could be further refined to account for complexity and unbalance of document structures and lexicon of actionable zones.

The estimated and simulated average number of user actions are presented on Figure 5. It shows that, as expected, the number of clicks (user actions) required to access an actionable zone is on average smaller for small documents with KeySurf, but much better and stable with 1Click for larger documents.

6. FUTURE WORK

So far we have made the simplifying assumption that a user is equally likely to reach for any element in the page, when in reality web designer strategically place the most attractive elements either at the beginning of the document to ensure usability, or at the end to encourage users to reads everything. In terms of usability, we argue that scanning based navigation is more intuitive than a search based navigation. The user does not need to switch his/her focus between the page and the on screen keyboard. Instead, the user only need to concentrate on where the focus of the scanner is and select the desired group of elements. Future work will focus on user studies with users needing such assistance, whether or not they currently use such assistive technologies.

The plugin is freely available on request to the authors. An immediate extension will be to further break up a long list of elements into smaller groups (ie. the language selection part in a Wikipedia page with 20+ links).

The technology we propose here could be combined with low resolution eye tracking to speed up access to large zones, and use the scan for parts of the page too small for high precision detection (such as a list of links for example).

7. ACKNOWLEDGEMENTS

Lars Gullestrup has contributed to the implementation.

8. REFERENCES

[1] J. Mankoff, A. Dey, U. Batra, and M. Moore. Web accessibility for low bandwidth input. In *In Proc. of the Fifth ACM SIGCAPH Conference on Assistive Technologies*, pages 17–24. ACM Press, 2002.

[2] J. Owens, S. Keller, and D. University. *MultiWeb [electronic resource] : Australian contribution to web accessibility / Janet Owens, Susan Keller*. Institute of Disability Studies, School of Management Information Systems Burwood, Vic, 2000.

[3] L. E. Sibert and R. J. K. Jacob. Evaluation of eye gaze interaction. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI00)*, pages 281–288, 2000.

[4] L. Spalteholz. Keysurf-a keyboard web navigation system for persons with disabilities. Master's thesis, University of Victoria, 2012.

[5] C. Stephanidis, A. Paramythis, C. Karagiannidis, and A. Savidis. Supporting interface adaptation: the avanti web-browser. In *Proc. of 3rd ERCIM workshop on user interfaces for all*, pages 3–4, 1997.